Computer-Aided Design 43 (2011) 1468–1476

Contents lists available at SciVerse ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Efficiently computing geodesic offsets on triangle meshes by the extended Xin–Wang algorithm

Shi-Qing Xin, Xiang Ying, Ying He *

School of Computer Engineering, Nanyang Technological University, Singapore

ARTICLE INFO

Keywords:
Discrete geodesics
Geodesic offsets
Geodesic distance field
Exact algorithm

ABSTRACT

Geodesic offset curves are important for many industrial applications, such as solid modeling, robot-path planning, the generation of tool paths for NC machining, etc. Although the offset problem is well studied in classical differential geometry and computer-aided design, where the underlying surface is sufficiently smooth, very few algorithms are available for computing geodesic offsets on discrete representation, in which the input is typically a polyline curve restricted on a piecewise linear mesh. In this paper, we propose an efficient and exact algorithm to compute the geodesic offsets on triangle meshes by extending the Xin–Wang algorithm of discrete geodesics. We define a new data structure called parallel-source windows, and extend both the "one angle one split" and the filtering theorem to maintain the window tree. Similar to the original Xin–Wang algorithm, our extended algorithm has an $O(n)$ space complexity and an $O(n^2 \log n)$ asymptotic time complexity, where n is the number of vertices on the input mesh. We tested our algorithm on numerous real-world models and showed that our algorithm is exact, efficient and robust, and can be applied to large scale models with complicated geometry and topology.

1. Introduction

Geodesic offsets or parallels are a set of points on the surface, which are at a constant geodesic distance from the generator curves. Computation of geodesic offsets plays an important role in many applications of computer-aided design, such as solid modeling [1,2], robot path planning [3], generation of tool paths for NC machining [4–6], etc.

Although the offset problem is well studied in classical differential geometry and computer-aided design, where the underlying surface is sufficiently smooth (e.g., [7–9]), very few algorithms are available for the computation of discrete geodesic offsets. Given a polygonal mesh $\mathcal{S}$ and a set of points and polylines, denoted by $\mathcal{U}$, that are restricted on $\mathcal{S}$, the distance-d geodesic offset consists of the points on $\mathcal{S}$ that are at a geodesic distance d from $\mathcal{U}$.

Holla et al. [10] proposed an incremental approach to approximate polyline-source offsets based on an assumption that the offset to a polyline on the surface is still a polyline. Chen et al. [11] computed approximate offsets by taking the samples on the source curve as source points. Bommes and Kobbelt [12] extended Mitchell et al.'s algorithm [13] to compute the polyline-source geodesic distance field. However, these algorithms are either approximate or computationally expensive/memory inefficient, which are hard to apply onto large-scale real-world models.

In this paper, we propose an *efficient* and *exact* algorithm to compute geodesic offsets on triangle meshes. Our algorithm extends the Xin–Wang algorithm of "single-source-all-destination" discrete geodesics [14] to the general geodesic offset problem where the generator is a set of points and polylines restricted on the input triangle mesh. Our algorithm has an $O(n)$ space complexity and an $O(n^2 \log n)$ asymptotic time complexity, where n is the number of vertices on the input mesh. Numerous experimental results show that our offset algorithm runs very fast in practice. As shown in Fig. 1, it takes 3.2 s and 3.06 MB memory to compute exact geodesic offsets on the Happy Buddha with 100K vertices.

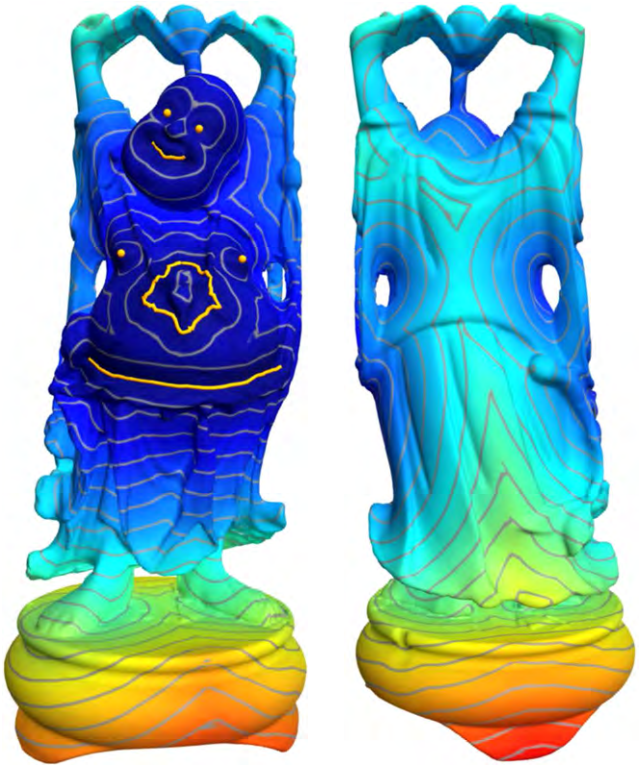
The rest of the paper is organized as follows. Section 2 reviews the related work in discrete geodesics and geodesic offsets. Section 3 briefly describes the Xin–Wang algorithm of "single-source-all-destination" discrete geodesics. Section 4 documents the extension of Xin–Wang algorithm for computing geodesic offsets on triangle meshes, followed by experimental results and discussions in Section 5. Finally, Section 6 concludes the paper.

2. Related work

Point-source geodesic distance field. Geodesic offsets are closely related to the computation of point-source geodesic distance field from which the geodesic offsets can be easily extracted. Sharir and Schorr [15] pioneered an algorithm to compute "single-source-all-destination" discrete geodesics on convex polyhedra with an

* Corresponding author. Tel.: +65 6514 1008.
E-mail addresses: sqxin@ntu.edu.sg (S.-Q. Xin), ying0008@e.ntu.edu.sg (X. Ying), yhe@ntu.edu.sg (Y. He).

0010-4485/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2011.08.027



(a) Front view. (b) Back view.

Fig. 1. User specifies 4 generator points and 3 generator curves (drawn in yellow) on the Happy Buddha with 100K vertices. Our algorithm computes the exact geodesic offsets in 3.2 s with 3.06 MB memory usage. The timing is measured on a DELL Workstation with a Xeon 2.66 GHz CPU and 4 GB RAM. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

$O(n^3 \log n)$ time complexity. Later, Mitchell et al. [13] (MMP) improved the time complexity bound to $O(n^2 \log n)$ by using the "continuous Dijkstra" technique. Chen and Han [16] (CH) suggested building a binary tree to encode all the edge sequences that can possibly contain a shortest path, thereby improving the time complexity to $O(n^2)$. Surazhsky et al. [17] extended the MMP algorithm to compute approximate geodesics with bounded error. Recently, Xin and Wang [14] improved the CH algorithm by exploiting a filtering theorem. Their improved algorithm outperforms both the MMP and CH algorithms. Besides the exact algorithms, there are also many algorithms [18–27] to approximate the discrete geodesic distance field. Among them, the fast marching method [23], with an $O(n \log n)$ time complexity, is a numerical method for solving boundary value problems of the Eikonal equation and has been widely used in the research community.

Geodesic path. Geodesic offsets are also related to geodesic paths since geodesic paths are perpendicular to geodesic offsets with the same source set. The ϵ-approximation algorithms [28–30] are well studied to compute an approximate path at most $(1+\epsilon)d(s,t)$ in length, where $d(s,t)$ denotes the exact geodesic distance between two points s and t on the input polyhedral surface. These algorithms make a trade-off between computation cost and accuracy. Polthier and Schmies [31] introduced discrete geodesic curvature on polyhedral surfaces and defined discrete straightest geodesics that allow a unique geodesic of the initial value problem. Later, they developed a method to compute the evolution of distance circles on polyhedral surfaces using geodesic flow [32]. Xin and Wang [33] presented a fast algorithm to compute a locally exact geodesic between two distinct vertices on a polyhedral surface. Recently, Xin et al. [34] proposed an efficient iterative method for computing exact geodesic loops within finite iterations.

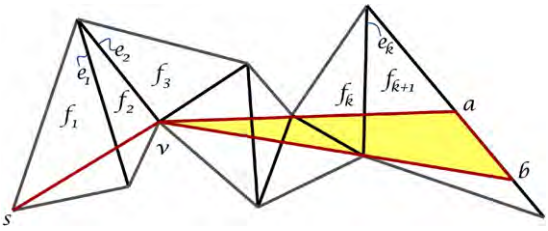


Fig. 2. All the shortest paths from s to any point $p \in [a, b]$ pass through the same edge sequence $(e_1, e_2, \ldots, e_k)$. Therefore we can use a funnel-like interval window to encode all such shortest paths.

Geodesic offsets. There is a rich body of literature of computing geodesic offsets on parametric surfaces. We refer the readers to the comprehensive surveys [35–37]. However, very few algorithms are available for computing discrete geodesic offsets on polygonal meshes. Holla et al. [10] computed offsets at distances $d_1, d_2, \ldots, d_k$ from the source curve, where $d_i < d_{i+1}$, $i = 1, 2, \ldots, k$. For any input $d \in [d_j, d_{j+1}]$, the geodesic offset can be reported by interpolating the offsets at d_j and d_{j+1}. However, their algorithm heavily depends on the assumption that the offset to a polyline is still a polyline. In fact, the assumption only holds on convex polyhedral surfaces. Chen et al. [11] took a set of sample points on the source curve as source points and then computed the geodesic offset at a given distance. Both [10,11] are approximate algorithms that require interactive detection and correction of global self-intersections. Bommes and Kobbelt [12] extended the MMP algorithm [13] onto the polyline-source geodesic offset problem. However, due to the high memory cost $O(n^2)$, their method is not applicable for large-scale models.

3. The Xin–Wang algorithm of discrete geodesics

This section briefly describes the key concepts and techniques in the Xin–Wang algorithm of "single-source-all-destination" discrete geodesics [14], which will be extended to solve the discrete geodesic offsets in the next section. We begin with several definitions for *edge sequences* and *windows* followed by the outline of the Xin–Wang algorithm.

3.1. Edge sequences and windows

Face sequences, edge sequences and windows are fundamental concepts in discrete geodesic algorithms [13–17].

A *face sequence* $\mathcal{F}$ is an ordered list of faces, say $f_1, f_2, \ldots, f_{k+1}$, such that each pair of consecutive faces, i.e., f_i and f_{i+1}, share a common edge, say e_i. The associated list of shared edges, say $\mathcal{E} = (e_1, e_2, \ldots, e_k)$, is called an *edge sequence*. We say that $\mathcal{E}$ is *simple* if it contains no repeated edges. Fig. 2 gives an example of edge sequence.

Planar unfolding is a widely used technique for computing shortest paths on polygonal meshes. A face sequence $\mathcal{F}$ can be unfolded in this way: rotate f_2 around e_1 to make f_1 and f_2 coplanar; rotate f_3 around e_2 to make f_2 and f_3 coplanar; and repeat this until all faces are coplanar. Obviously, this process takes $O(k)$ time. The detailed technique is available in [17].

Mitchell et al. [13] observed that any shortest path passes through an alternating list of vertices and edge sequences. Surazhsky et al. [17] used a structure *window* to encode the points whose shortest paths share the same vertex–edge sequence. In Fig. 2, the last vertex v in this vertex–edge sequence is called a *root*

or *pseudo-source*, which is crucial to define a window since $d(s, p)$ equals to the sum of $d(s, v)$ and the straight-line distance between the unfolded image point of v and p, where $d(\cdot, \cdot)$ returns the geodesic distance. We can further classify windows into *pseudo-source windows* that end at a vertex and *interval windows* that end at a non-empty interval of an edge.

3.2. Window derivation

Unlike the graph-based shortest path problem, the discrete geodesic problem on a polyhedral surface is continuous in essence. The key to discretize the geodesic problem is to encode the shortest paths sharing the common vertex–edge sequence into a window and then take care of the derivation of windows. Most of the existing exact geodesic algorithms [13,14,16] compute the children of an existing window w in a similar fashion (see Fig. 3):

If *w is a pseudo-source window at vertex v with geodesic distance d*
  If *v is a saddle vertex* (*see* Fig. 3(a))
    For *each adjacent vertex, compute a pseudo-source window*;
    For *each opposite edge, compute an interval window*;
  Else */*v is a convex vertex*/*
    No children need to be computed;
Else */*w is an interval window. Suppose w covers an interval $[a, b]$ of the edge $\overline{v_1v_2}$. Let v be the vertex opposite to $\overline{v_1v_2}$ and I be the unfolded image of the last vertex passed through by w.*/*
  If *the line segment $\overline{Iv}$ is right to the interval $[a, b]$* (*see* Fig. 3(b))
    Compute the only interval-window child on edge $\overline{v_1v}$;
  Elseif *the line segment $\overline{Iv}$ is left to the interval $[a, b]$* (*see* Fig. 3(c))
    Compute the only interval-window child on edge vv_2;
  Else */*$\overline{Iv}$ intersects the interval $[a, b]$ at some point* (*see* Fig. 3(d)).*/*
    Compute two interval windows as the children, one on edge v_1v and the other on edge vv_2;
    Compute a pseudo-source window at vertex v.

3.3. Two key observations

The existing geodesic algorithms [13,16,14] differ in the techniques to avoid the combinatorial explosion of the number of windows. The MMP algorithm [13] transforms each edge into a covering of a set of ordered windows. Chen and Han [16] (CH) observed an important fact called "one angle one split", with which they built an $O(n^2)$ tree to encode the windows that possibly determine a shortest path. Xin and Wang [14] observed that the majority of windows created by the CH algorithm are useless and presented a filtering theorem that significantly reduces the number of windows.

One angle one split. As shown in Fig. 4(a), w_1 and w_2 are two interval windows that cover the same angle $\angle v_1vv_2$. According to the window derivation algorithm in Section 3.2, both w_1 and w_2 may have two children, which leads to four new windows w_1^1, w_1^2, w_2^1, w_2^2. However, Chen and Han's "one angle one split" theorem [16] states that at least one of the four children does not help when determining shortest paths. That is to say, among all the windows covering the same angle, at most one of them can have two children that possibly determine a shortest path. Thus, one associates each angle with a "winning" window, by which one can check the validity of new windows. The CH algorithm guarantees that the window tree is $O(n^2)$ in size and $O(n)$ in depth.

Filtering theorem. Xin and Wang [14] observed that the majority of windows created by the CH algorithm [16] are useless. Therefore they proposed a filtering theorem to remove those useless windows. As shown in Fig. 4(b), w is an interval window on the edge $\overline{v_1v_2}$ and the unfolded image of w's root is located on the plane of $\triangle v_1v_2v_3$. The filtering theorem asserts that w is useless if one of the following conditions holds:

- The shortest path from s to v_1 and then to B is shorter than what w can give;
- The shortest path from s to v_2 and then to A is shorter than what w can give;
- The shortest path from s to v_3 and then to A is shorter than what w can give.

3.4. Algorithm review

Besides the two key observations, the Xin–Wang algorithm [14] also suggested maintaining a priority queue throughout the algorithm such that the "nearest" window can be handled first. In the following, we review the Xin–Wang algorithm using pseudo-code.

Initialization: (1) *Set the geodesic distance of the source points to be zero and that of other vertices to be infinity*; (2) *Create a priority queue $\mathcal{Q}$ to contain the pseudo-source windows rooted at the source points; and* (3) *For each directed edge, we associate it with a NULL interval window for the purpose of calling "one angle one split".*
While $\mathcal{Q}$ is not empty
  Pop out the head element w in $\mathcal{Q}$;
  If *w is a pseudo-source window at v with distance d*
    If *d is less than the current distance estimate at v*
    Update the geodesic distance of v to be d;
    If *v is a saddle vertex*
      */*Let w_{old} be old pseudo-source window at v.*/*
      Delete the subtree rooted at w_{old};
      Associate v with w;
      For *each vertex neighboring to v*,
        Create a child pseudo-source window;
        Push it into $\mathcal{Q}$;
      For *each edge opposite to v*
        Create a child interval window;
        Push it into $\mathcal{Q}$;
  Else */*w is an interval window at edge e. Let v_e be the vertex opposite to e and w_{old} be the old window associated with e.*/*
    If *w covers v_e and can provide a shorter distance for v_e than w_{old}*
      Update w_{old} to be w;
      Delete one subtree of w_{old};
      Compute w's two children w_1 and w_2;
      Check w_1 and w_2 with the filtering theorem;
      Push the children into $\mathcal{Q}$ if they passed check;
      Update the distance at v_e if w can provide v_e with a shorter distance;
    Else */*w can have only one child.*/*
      Compute the only child;
      Check it with the filtering theorem;
      Push the child window into $\mathcal{Q}$ if it is valid.

Numerous experimental results show that the Xin–Wang algorithm [14] outperforms the MMP algorithm [13] and the CH algorithm [16] in both time and space. This motivates us to extend the Xin–Wang algorithm onto the general geodesic offset problem.

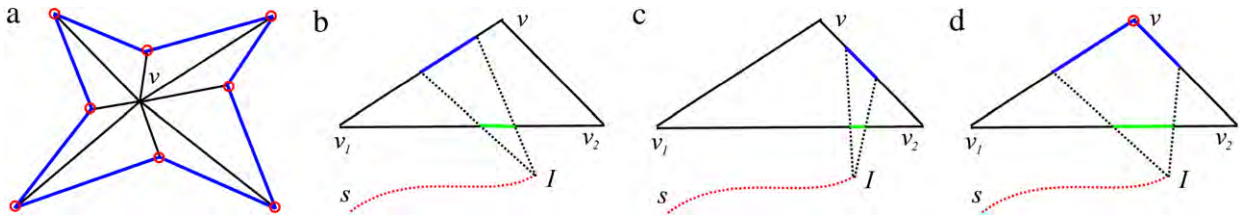


Fig. 3. Computing the children of a window: (a) A pseudo-source window at a saddle vertex v may have children both on opposite edges and at adjacent vertices; (b–d) An interval window w on edge $\overline{v_1v_2}$ may have one or two interval-window children, depending on how the line segment $\overline{Iv}$ intersects w's interval.

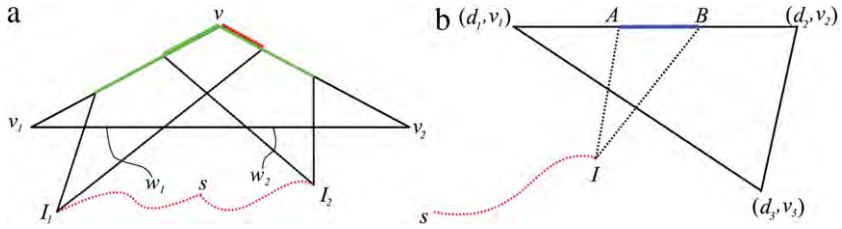


Fig. 4. Section 3.3—Two key observations in the Xin–Wang algorithm [14]: (a) Chen and Han's "one angle one split" [16] states that at most one of the windows covering the same angle can have two children; and (b) Xin and Wang's filtering theorem [14] asserts that an interval window is useless if it cannot provide a shorter geodesic distance than one of the neighboring vertices.

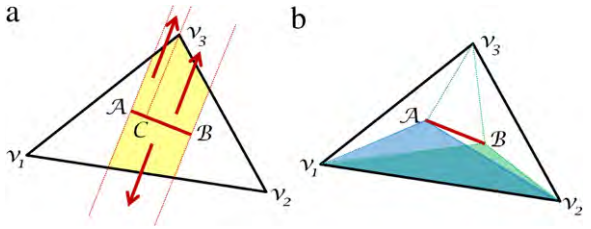


Fig. 5. Three types of windows, i.e., pseudo-source windows, interval windows and parallel-source windows are required to compute the general offset problem. (a) Taking the line segment AB as the source, we need to construct three parallel-source windows initially. (b) We also need to construct 6 interval windows rooted at the endpoints A and B.

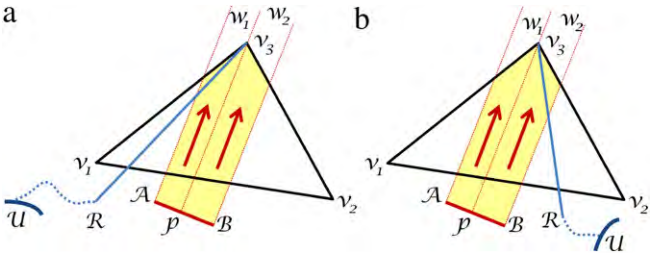


Fig. 6. Extended "one angle one split": if the parallel-source window w can provide a shorter distance to v_3 than the existing winning window, then we allow w to have two children. Otherwise, we remove either w_1 (see (a)) or w_2 (see (b)).

4. Extending the Xin–Wang algorithm to the general geodesic offset problem

This section details our algorithm to compute the general geodesic offset on triangle meshes. The key components of our algorithm are to define a new type of windows called *parallel-source windows* and then extend Chen and Han's "one angle one split" and Xin and Wang's filtering theorem (see Fig. 4).

4.1. Parallel-source window

Consider the source set $\mathcal{U}$ including a collection of source points and polyline curves. After computing the geodesic distance field, we can backtrace the discrete geodesic, for any destination point p, which is actually the shortest path from p to $\mathcal{U}$. We use $\mathcal{G}$ to represent the family of backtraced discrete geodesics and $\mathcal{O}$ the family of offset curves. In the continuous setting, the offset curves are iso-distance curves and the backtraced geodesics follow the gradients of the geodesic distance field. Thus, each curve in $\mathcal{G}$ is orthogonal to a curve in $\mathcal{O}$, i.e., $\forall \mathcal{C}_1 \in \mathcal{G}$, $\mathcal{C}_2 \in \mathcal{O}$, $\mathcal{C}_1 \perp \mathcal{C}_2$.

On triangle meshes, the above condition implies that each of the four angles around the intersection point between $\mathcal{C}_1$ and $\mathcal{C}_2$ is greater than or equal to $\pi/2$. Specially, for the case of Fig. 5(a) where the source is the line segment AB, we need to consider the geodesics orthogonal to AB besides the geodesics rooted at A or B. Therefore, we define another window type, i.e., *parallel-source* windows for purpose of encoding a set of shortest paths that share the same edge sequence and are orthogonal to the source segment.

Taking Fig. 5 as an example, we need to push 3 parallel-source windows (see Fig. 5(a)) and 6 interval windows rooted at A and B (see Fig. 5(b)) into a priority queue when initializing the Xin–Wang algorithm [14].

4.2. Extending Chen and Han's "one angle one split"

Chen and Han's "one angle one split" [16] states that at most one of the windows covering the same angle can have two children (see Fig. 4(a)). In implementation, we need to keep a "winning" window that can provide the up-to-date shortest geodesic distance for the opposite vertex. In this subsection, we discuss how to extend "one angle one split" onto the general offset problem.

As shown in Fig. 6, we assume that v is the *last* vertex passed through by the existing shortest geodesic path between s and v_3, and the unfolded image of v is R. Since any window encodes a set of shortest paths sharing the same edge sequence, there is no vertex enclosed in the yellow area in Fig. 6(a). Therefore, R must be outside of the yellow area. Then we consider the following cases in the extended "one angle one split":

Extended "one angle one split":

Case 1. *If $p \in AB$ can give a shorter distance to v_3 to than the existing winning window w_{old}, then the parallel-source window w is allowed to have two children; and the winning window of v_3 is set to be w.*

Case 2. *If $p \in AB$ cannot give a shorter distance to v_3,*

Case 2.1. *If Rv_3 is left to AB, then we remove the left child w_1; see Fig. 6(a).*

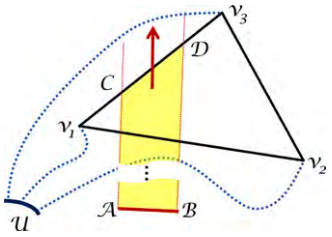


Fig. 7. The parallel-source window cannot contribute to determining a shortest path if v_1 gives a shorter distance than $\|BD\|$, or v_2 gives a shorter distance than $\|AC\|$, or v_3 gives a shorter distance than $\|AC\|$.

Case 2.2. *If Rv_3 is right to AB, then we remove the right child w_2; see* Fig. 6(b).

Remarks. (1) Note that if Rv_3 intersects AB at an interior point q, it can be shown that the new window can give a shorter distance than w_{old}, which belongs to Case 1. (2) If w_{old} is also a parallel-source window, then the root of w_{old} is defined to be the point in the source segment of w_{old} that is nearest to v_3.

4.3. Extending Xin and Wang's filtering theorem

As shown in Fig. 7, the side boundaries AC and BD of a parallel-source window are orthogonal to the source segment AB. Therefore, A (resp. B) is nearer to C (resp. D) than any other point in AB. This motivates us to extend Xin and Wang's filtering theorem [14]:

Extended filtering theorem. Suppose the parallel-source window w arrives at the face $\triangle v_1v_2v_3$, entering from v_1v_2 and exiting from v_1v_3 (see Fig. 7*). Let AB be the unfolded image of the source segment and CD be the interval in v_1v_3 covered by w. Then the parallel-source window w can be filtered out (i.e., it does not help to determine a shortest path) if*

$$d(\mathcal{U}, v_1) + \|v_1D\| \le \|BD\|, \tag{1}$$

or

$$d(\mathcal{U}, v_3) + \|v_3C\| \le \|AC\|, \tag{2}$$

or

$$d(\mathcal{U}, v_2) + \|v_2C\| \le \|AC\|, \tag{3}$$

where $d(\mathcal{U}, v_i)$ is the geodesic distance from v_i to the nearest point in $\mathcal{U}$, while $\|\cdot\|$ denotes the straight-line distance.

Proof. We consider three cases:

Case 1. If v_1 can provide a shorter distance for the point D, i.e.,

$$d(\mathcal{U}, v_1) + \|v_1D\| \le \|BD\|,$$

then for any point $p \in CD$, the geodesic distance given by v_1 is

$$d(\mathcal{U}, v_1) + \|v_1p\|,$$

while the geodesic distance given by the parallel window w is $\|pp'\|$, where p' is the projection point on the segment AB (see Fig. 8). Considering

$$\|BD\| - \|pp'\| < \|pD\|,$$

we have

$$\begin{aligned} d(\mathcal{U}, v_1) + \|v_1p\| &= d(\mathcal{U}, v_1) + \|v_1D\| - \|pD\| \\ &< (\|BD\|) - (\|BD\| - \|pp'\|) \\ &= \|pp'\|. \end{aligned} \tag{4}$$

Observing that inequality (4) holds for any $p \in CD$, we can show that the parallel-source window w is useless and can be filtered out.

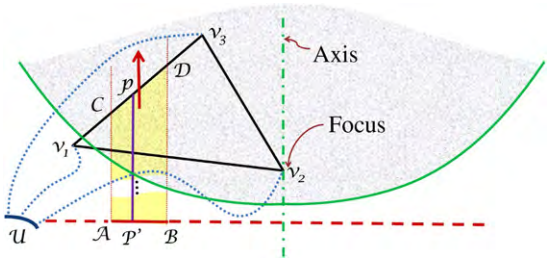


Fig. 8. Case 3.

Case 2. If v_3 can provide a shorter distance for C, i.e.,

$$d(\mathcal{U}, v_3) + \|v_3C\| \le \|AC\|,$$

we can prove that the window w is filtered out in a similar fashion as Case 1.

Case 3. If v_2 can provide a shorter distance for C than that of the parallel-source window w, we can easily show that both v_2 and C are on the same side of the straight line AB (as shown in Fig. 8). We construct a parabola $\mathcal{P} \in \mathbb{R}^2$ such that

$$d(\mathcal{U}, v_2) + \|x - v_2\| = \|xx'\|,$$

where x is a moving point on the plane of $\triangle v_1v_2v_3$ and x' is the projection point of x on the straight line AB. Obviously, C is located in the interior of $\mathcal{P}$ due to the assumption

$$d(\mathcal{U}, v_2) + \|v_2C\| \le \|AC\|.$$

Then we show that D must be in the interior of $\mathcal{P}$ by contradiction. Assume D is outside of $\mathcal{P}$. D must be on the right of the axis of $\mathcal{P}$ because

(i) D is right to C along $\overrightarrow{AB}$, i.e., $\overrightarrow{AB} \cdot \overrightarrow{CD} > 0$;
(ii) D is above the straight line v_1v_2 (BD intersects v_1v_2 at one interior point), where v_2 is the focus of $\mathcal{P}$.

But on the other hand, D cannot be right to the axis of $\mathcal{P}$ because the window w enters the triangle $\triangle v_1v_2v_3$ from v_1v_2 and leaves it from v_1v_3.

With the fact that D is in the interior of $\mathcal{P}$, the entire segment CD is in the interior of $\mathcal{P}$ due to the convexity. This implies

$$d(\mathcal{U}, v_2) + \|v_2p\| \le \|pp'\|$$

for any $p \in CD$, where p' is the projection point of p onto AB. Therefore, the parallel-source window w can be filtered out.

Putting all the three cases together, we complete the proof. □

4.4. Complexity analysis

Our extended Xin–Wang algorithm applies to the general geodesic offset problem, where the generators include both points and polylines. For simplicity, we assume that the generator points form a subset of the mesh vertices[1] and each mesh triangle contains at most one line segment of the generator curves (polylines), whose endpoints serve as the entry point and the exit point respectively. It is clear that we have $O(m)$ parallel-source windows initially, where m is the number of faces ($m \approx 2n$ for triangle meshes). The property "one angle one split" guarantees

[1] For the general case where the generator point is inside a triangle but not a mesh vertex, one can simply split the triangle into three sub-triangles such that the generator point becomes a mesh vertex.

Fig. 9. Experimental results. The generator curves are drawn in yellow. Cold (resp. warm) colors mean small (resp. large) geodesic distances. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

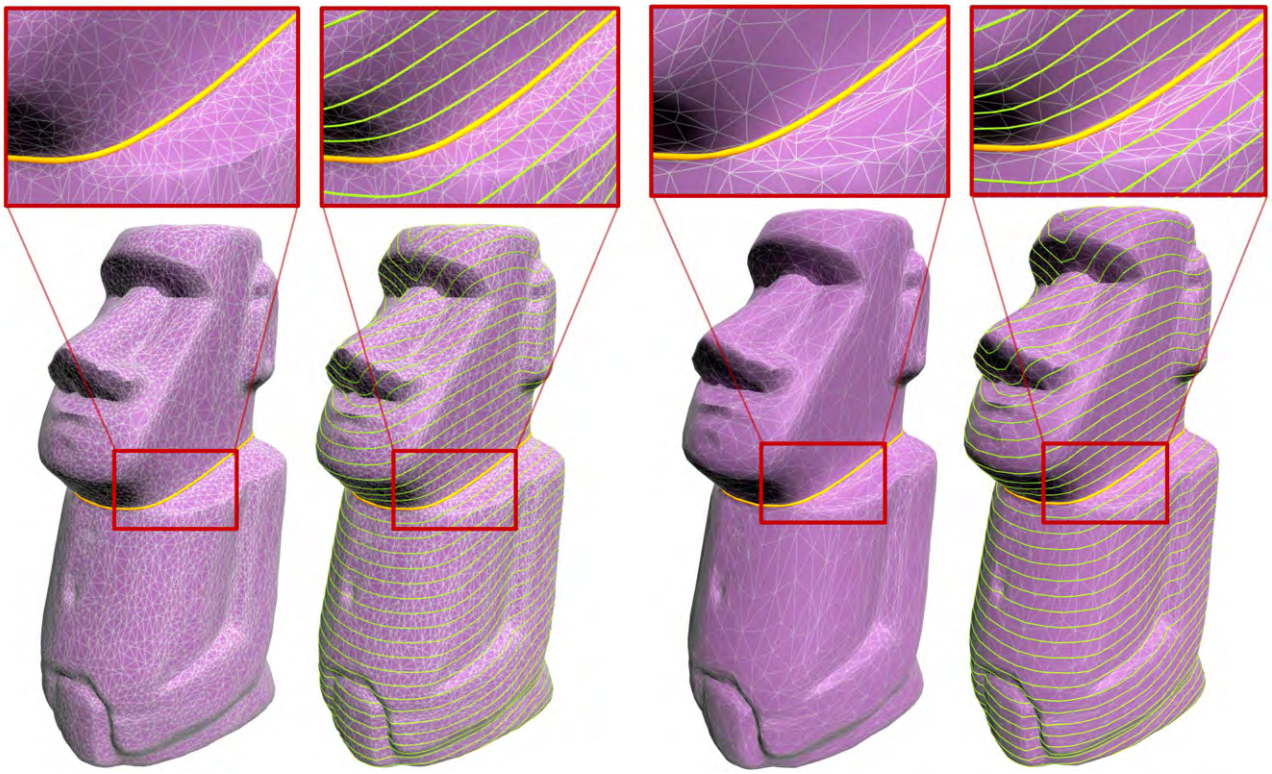


Fig. 10. Our geodesic offset algorithm is intrinsic to the geometry and insensitive to the mesh resolution and triangulation. Left: the Moai model with 11K vertices; Right: the Moai model with 3K vertices.

that the number of windows on each level can be bounded by $O(m)$ when building the window tree [16]. Since the window tree cannot exceed m levels (a shortest path intersects every face at most once), the total number of windows is $O(m^2)$. Note that we only need to maintain the bottom level (i.e., leaves) of the windows tree and all branch nodes. Therefore, the extended algorithm has still an $O(n)$ space complexity and an $O(n^2 \log n)$ asymptotic time complexity as the original Xin–Wang algorithm [14].

5. Experimental results

This section documents the experimental results and demonstrates that our algorithm is efficient, exact, intrinsic and robust. We implemented the extended Xin–Wang algorithm in C++ on Windows 7. Our experiments were carried out on a DELL Workstation with a Xeon 2.66 GHz CPU and 4 GB RAM.

Table 1
Timing statistics are measured in seconds on a workstation with a Xeon 2.66 GHz CPU and 4 GB RAM.

Models	#vertices	Timing (s)	Memory (MB)
Armadillo (Fig. 9)	173*K*	13.385	5.29
Buddha (Fig. 1)	100*K*	3.224	3.06
Bunny (Fig. 9)	75*K*	8.537	2.23
Dragon (Fig. 9)	750*K*	77.173	22.90
Gargoyle (Fig. 9)	352*K*	31.840	10.76
Laurana (Fig. 14)	10*K*	0.733	0.33
Lucy (Fig. 11)	263*K*	26.114	8.04
Moai (Fig. 10 left)	11*K*	0.655	0.34
Moai (Fig. 10 right)	3*K*	0.06	0.08
Two kids (Fig. 13)	42*K*	3.931	1.29

Fig. 11. Given the generator curve on the waist, we compute the geodesic distance field of the Lucy model. Here we show the set of offset curves with spacing of 0.06 (left) and 0.025 (right) respectively. The model is normalized to a unit cube.

5.1. Performance

We tested our algorithm on numerous commonly-used models in graphics and geometric modeling community. Due to the $O(n)$ space complexity and $O(n^2 \log n)$ time complexity, our algorithm works efficiently and effectively for large scale models (see Table 1 for the statistics). Taking the Dragon model (with 750*K* vertices) as an example, our algorithm computes the geodesic offsets for the "SPM11" logo within only 77 s and costs 23 MB memory. Fig. 11 shows the geodesic parallels on the Lucy model with various spacings. Fig. 12 demonstrates the geodesic offset on the 3D scanned human face model, which usually contains artifacts (such as disconnected components, holes, etc.) near the face boundary. Computing the geodesic offset based on the generator curves of some facial features (e.g., eyes, nose and mouth) leads to a good segmentation.

Fig. 12. Taking the salient features (e.g., eyes, forehead, nose and mouth) and shortest paths between them as generators, we compute the geodesic offsets, which can be used for face segmentation.

5.2. Robustness

The geodesic computation is intrinsic to the geometry and insensitive to the mesh resolution and tessellation. As shown in Fig. 10, we tested our algorithms on the Moai model with different resolutions and triangulations. The generator curves, specified by the same cutting plane, are the same for both experiments. Our algorithm is stable and robust, which produces consistent resultant geodesic offsets.

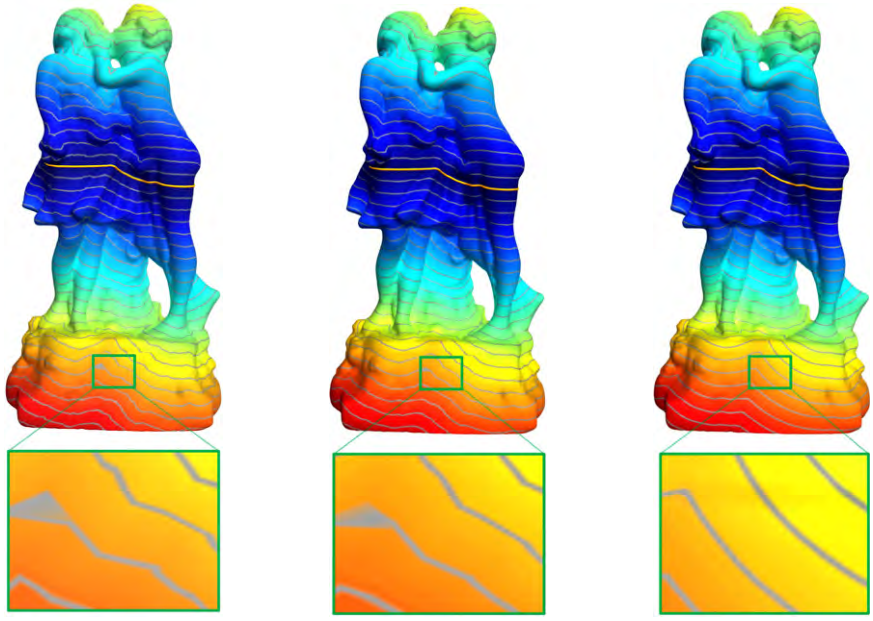
5.3. Exactness

The proposed geodesic offset algorithm is exact in the sense that the geodesic distance from each vertex to the generator curve(s) is accurate. As a result, the integral curves of the gradient of the geodesic distance field are exact geodesics. As shown in Fig. 14, the (blue) geodesics are orthogonal to the (green) offset curves everywhere.

To demonstrate the exactness of the proposed algorithm, we also compared our algorithm with Dijkstra's algorithm [38] and the fast marching method [23], which are *approximation* algorithms for geodesic offsets. As shown in Fig. 13, the approximation algorithms usually result in poor quality geodesic offsets on models with complicated geometry and topology, while our result is much better. This is because the geodesic offsets are G^1 continuous except at conjugate points (that have at least two different shortest paths to the source). We must point out that the overhead for the exactness is the high computational cost: both Dijkstra's algorithm [38] and the fast marching method [23] are of time complexity $O(n \log n)$, while ours is $O(n^2 \log n)$.

5.4. Discussions

Holla et al. [10] and Chen et al. [11] used a traditional approach to compute geodesic offsets, where the offsets are computed



(a) Dijkstra [38] 0.078 s. (b) Fast marching [23] 0.31 s. (c) Our method 3.93 s.

Fig. 13. Comparison to Dijkstra's algorithm [38] and the fast marching method [23]. Note that our algorithm is able to compute the *exact* geodesic offsets while the others compute only the *approximate* geodesic offsets (see the artifacts in (a) and (b)).

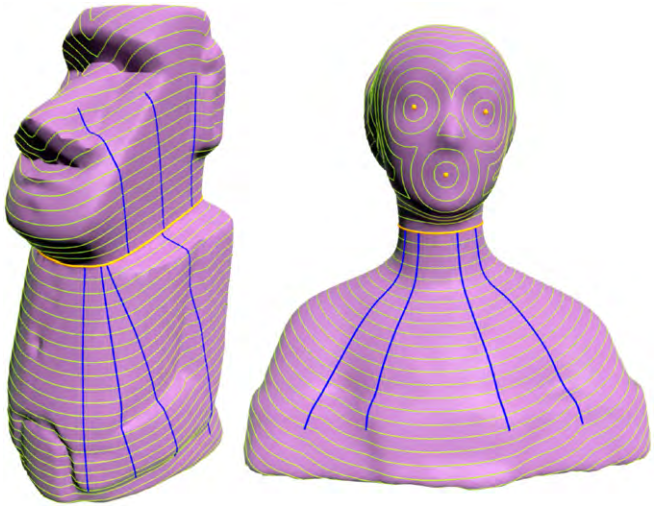


Fig. 14. Geodesics (drawn in blue) starting from the generator curves (drawn in yellow) are orthogonal to the offset curves (drawn in green) everywhere. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

segment by segment followed by local and global trimmings. It is known that this kind of approach is computationally expensive and can hardly be used for meshes with more than a few thousand vertices. Furthermore, it is technically challenging to fix global intersections. Usually, manual inspection and correction of self-intersections are required.

Bommes and Kobbelt [12] extended the MMP algorithm [13] onto the polyline-source geodesic offset problem. Similar to our algorithm, the extended MMP algorithm can compute the *exact* geodesic offset on polygonal meshes. Although working well for small and medium sized meshes, their approach is impractical for large scale models due to its high space complexity $O(n^2)$. For example, their algorithm failed on a mesh with $200K$ vertices by running out of memory on a PC with 2 GB RAM. By contrast, our algorithm successfully computed the exact geodesic offsets by using only 6 MB memory. To our knowledge, our algorithm is the first one that is able to compute *exact* geodesic offsets on large scale polygonal meshes. Besides, the MMP algorithm suggests transforming each edge into a covering of windows, which makes windows become very narrow during derivation. The proposed algorithm by Bommes and Kobbelt [12] also inherits the disadvantage and thus is error prone.

In addition, we observe that the mesh complexity is not the only factor for determining the time cost. Generally speaking, the multiple-source version of the Xin–Wang algorithm runs faster than the single-source version. Long running time is required for models with a concentrated configuration of conjugate points. Even if running the single-source Xin–Wang algorithm on the same model, different source settings will report different time costs. Anyway, complicated topology and complicated geometry do not imply high computation costs.

6. Conclusion and future work

We proposed an efficient algorithm to compute exact geodesic offsets on triangle meshes. Our algorithm extends the Xin–Wang algorithm of discrete geodesics by introducing parallel-source windows and generalizing both "one angle one split" and the filtering theorem, which are the key components in the original Xin–Wang algorithm. The extended algorithm has an $O(n)$ space complexity and an $O(n^2 \log n)$ asymptotic time complexity, where n is the number of vertices on the input mesh. Our algorithm outperforms the existing discrete geodesic offset algorithms in both timing and memory usage. We tested the proposed algorithm on numerous real-world models. The promising experimental results showed that our algorithm is exact, efficient, intrinsic and robust.

We believe that the proposed algorithm has great potential in real-world problems. We will investigate various applications in the near future. It would also be interesting to develop a GPU-based algorithm to further improve the performance.

Acknowledgments

This project was partially supported by the Singapore National Research Foundation Interactive Digital Media R&D Program, under research grant NRF2008IDM-IDM004-006. We would like to thank the anonymous reviewers for their constructive comments and Shuang-Min Chen for her help on figures. Models courtesy of Stanford University and Aim@Shape.

References

[1] Rossignac JR, Requicha AAG. Offsetting operations in solid modelling. Computer Aided Geometric Design 1986;(3):129–48.
[2] Forsyth M. Shelling and offsetting bodies. In: Proceedings of the third ACM symposium on solid modeling and applications. SMA'95. 1995. p. 373–81.
[3] Lozano-Perez T, Wesley M. An algorithm for planning collision free paths amongst polyhedral obstacles. Communications of the ACM 1979;25(9): 560–70.
[4] Marciniak K. Geometric modeling for numerically controlled machining. Oxford University Press; 1991.
[5] Choi BK, Jerard R. Sculptured surface machining—theory and applications. Kluwer Academic Publishers; 1998.
[6] Kim KI, Kim K. A new machine strategy for sculptured surfaces using offset surface. International Journal of Production Research 1995;33(6):1683–97.
[7] Patrikalakis NM, Bardis L. Offsets of curves on rational *b*-spline surfaces. Engineering with Computers 1989;5(1):39–46.
[8] Hatna A, Grieve RJ, Broomhead P. Offsetting 3D contours on parametric surfaces. The International Journal of Advanced Manufacturing Technology 2000;16(3):189–95.
[9] Piegl LA, Tiller W. Computing offsets of NURBS curves and surfaces. Computer-Aided Design 1999;31(2):147–56.
[10] Holla VD, Shastry KG, Prakash BG. Offset of curves on tessellated surfaces. Computer-Aided Design 2003;35(12):1099–108.
[11] Chen Z, Chen Y, Liang F. A new offset approach for curves on triangle mesh surfaces. In: Technology and innovation conference 2006. 2006. p. 202–8.
[12] Bommes D, Kobbelt L. Accurate computation of geodesic distance fields for polygonal curves on triangle meshes. In: VMV'07. 2007. p. 151–60.
[13] Mitchell JSB, Mount DM, Papadimitriou CH. The discrete geodesic problem. SIAM Journal on Computing 1987;16(4):647–68.
[14] Xin S-Q, Wang G-J. Improving Chen and Han's algorithm on the discrete geodesic problem. ACM Transactions on Graphics 2009;28: 104:1–104:8.
[15] Sharir M, Schorr A. On shortest paths in polyhedral spaces. SIAM Journal on Computing 1986;15(1):193–215.
[16] Chen J, Han Y. Shortest paths on a polyhedron. In: SCG'90: proceedings of the sixth annual symposium on computational geometry. 1990. p. 360–9.
[17] Surazhsky V, Surazhsky T, Kirsanov D, Gortler SJ, Hoppe H. Fast exact and approximate geodesics on meshes. ACM Transactions on Graphics 2005;24(3): 553–60.
[18] Hershberger J, Suri S. Practical methods for approximating shortest paths on a convex polytope in $\mathbb{R}^3$. In: SODA'95: proceedings of the sixth annual ACM–SIAM symposium on discrete algorithms. 1995. p. 447–56.
[19] Varadarajan KR, Agarwal PK. Approximating shortest paths on a nonconvex polyhedron. In: IEEE symposium on foundations of computer science. 1997. p. 182–91.
[20] Agarwal PK, Har-Peled S, Sharir M, Varadarajan KR. Approximating shortest paths on a convex polytope in three dimensions. Journal of the ACM 1997; 44(4):567–84.
[21] Lanthier M, Maheshwari A, Sack J-R. Approximating weighted shortest paths on polyhedral surfaces. In: SCG'97: proceedings of the thirteenth annual symposium on computational geometry. 1997. p. 485–6.
[22] Barbehenn M. A note on the complexity of Dijkstra's algorithm for graphs with weighted vertices. IEEE Transactions on Computers 1998;47(2):263.
[23] Kimmel R, Sethian JA. Computing geodesic paths on manifolds. Proceedings of the National Academy of Sciences USA 1998;8431–5.
[24] Har-Peled S. Approximate shortest paths and geodesic diameter on a convex polytope in three dimensions. Discrete & Computational Geometry 1999; 21(2):217–31.
[25] Agarwal PK, Har-Peled S, Karia M. Computing approximate shortest paths on convex polytopes. In: Symposium on computational geometry. 2000. p. 270–9.
[26] Spira A, Kimmel R. Geodesic curvature flow on parametric surfaces. In: Curve and surface design. 2002. p. 365–73.
[27] Aleksandrov L, Maheshwari A, Sack J-R. An improved approximation algorithm for computing geometric shortest paths. In: The 14th international symposium on fundamentals of computation theory. 2003. p. 246–57.
[28] Papadimitriou CH. An algorithm for shortest-path motion in three dimensions. Information Processing Letters 1985;20(5):259–63.
[29] Clarkson K. Approximation algorithms for shortest path motion planning. In: Proceedings of the nineteenth annual ACM symposium on theory of computing. 1987. p. 56–65.
[30] Choi J, Sellen J, Yap C-K. Approximate Euclidean shortest path in 3-space. In: Proceedings of the tenth annual symposium on computational geometry. 1994. p. 41–8.
[31] Polthier K, Schmies M. Mathematical visualization. Springer Verlag; 1998. [Chapter] Straightest geodesics on polyhedral surfaces, p. 391.
[32] Polthier K, Schmies M. Geodesic flow on polyhedral surfaces. In: Proceedings of eurographics-IEEE symposium on scientific visualization'99. 1999. p. 179–88.
[33] Xin S-Q, Wang G-J. Efficiently determining a locally exact shortest path on polyhedral surfaces. Computer-Aided Design 2007;39(12):1081–90.
[34] Xin S-Q, He Y, Fu C-W. Efficiently computing exact geodesic loops within finite steps. IEEE Transactions on Visualization and Computer Graphics, vol. 99. Los Alamitos, CA, USA: IEEE Computer Society, 2011 [in press].
[35] Pham B. Offset curves and surfaces: a brief survey. Computer-Aided Design 1992;24(4):223–9.
[36] Elber G, Lee I-K, Kim MS. Comparing offset curve approximation methods. IEEE Computer Graphics and Applications 1997;17(3):62–71.
[37] Maekawa T. An overview of offset curves and surfaces. Computer-Aided Design 1999;31(3):165–73.
[38] Dijkstra EW. A note on two problems in connexion with graphs. Numerische Mathematik 1959;1:269–71.